# Introduction:

This Python script implements an object distance estimator using the YOLOV8 model for object detection. Also uses OpenCV for video input, output processing and UI, the Ultralytics library for YOLO object detection, and pyttsx3 for text-to-speech functionality.

## Project Description:

Given a trained model to an object or multiple objects and this object's attributes as it's length, we can detect and get the estimated distance to this object from the camera's position.
And using some indicators to tell the user how to navigate the camera (in this case will be the ROV) to make the object in the center og the camera

## Project Features:

- Object Detection
- Distance to object estimator
- Text Indicator to show the user how to make the object be in the frame center (NEW)
- Voice Indicator to show the user how to make the object be in the frame center (NEW)
- Compass and Tracker Line to guide the user also to make the object be in the frame center (NEW)

## Code Explanation and Documentation:

### A-Class Estimator( ):

It's Constructor: `__init__(model_path, channel=0)`

- **Parameters:** - `model_path` : (string) Path to the YOLO model file. `channel` : the camera channel 0 default

#### Methods

#### 1-Private Method: `_GetFrameCenter()`

```
This function returns the center point of the frame
The center point is used in the new GetMovements Algorithm
```

#### 2-Private Method: `_DrawLine(frame, start_point, end_point, color=(0, 0, 255), thickness=2)`

```
This function draws a line on the given frame from `start_point` to `end_point`
with
specified color and thickness
Parameters: `frame`: Input frame on which the line is drawn
         `start_point`: Tuple (x, y) representing the starting point of the line
         `end_point`: Tuple (x, y) representing the ending point of the line
         'Other parameters used for coloring, thickness and more'
```

### 3-Private Method: `_DrawArrow(frame, direction, color=(0, 0, 255), thickness=2, max_length=50, padding=50)`

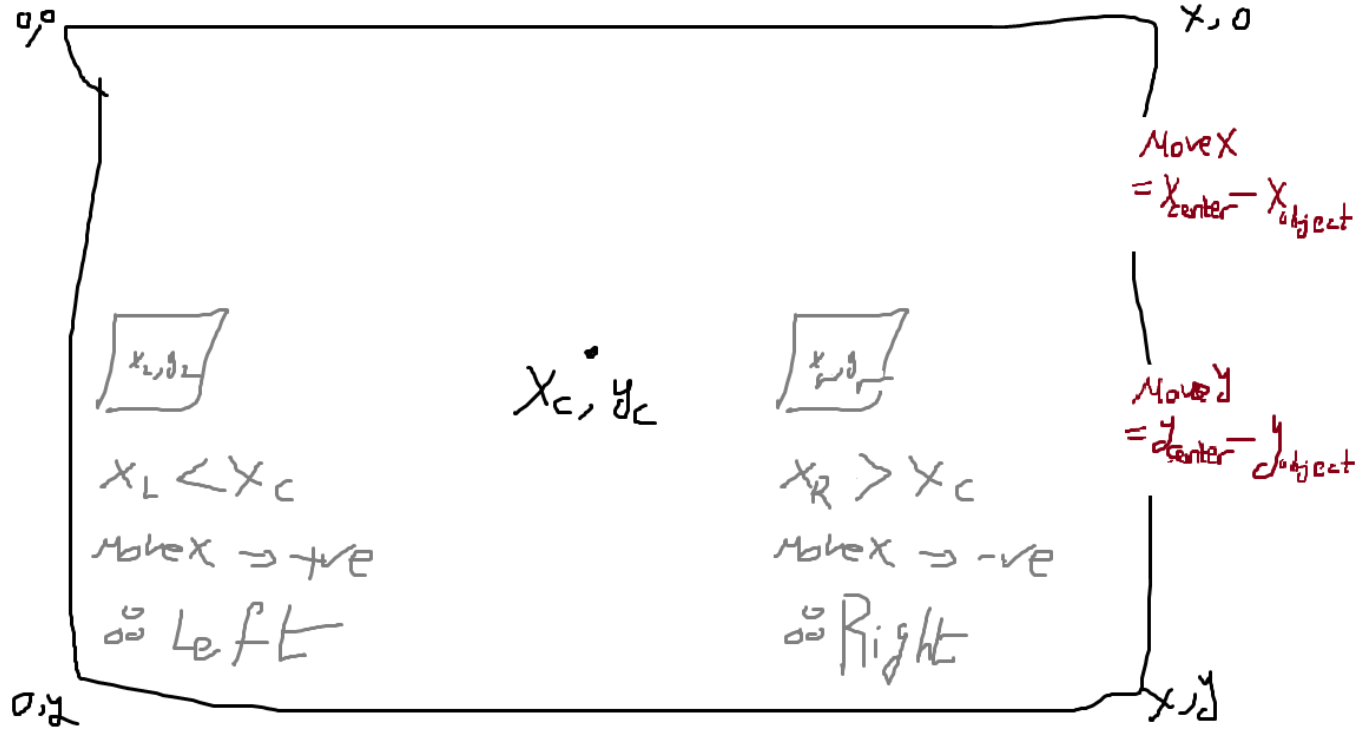
```
This function draws an arrow on the given frame indicating direction to move to
make
the object be in the center of the frame
 Parameters: `frame`: Input frame on which the arrow is drawn
        `direction`: Tuple (x, y) representing the direction vector.
         'Other parameters used for coloring, thickness and more'
```

### 4-Private Method: `_GetMovements(frame, frameX, frameY, boxX, boxY)`

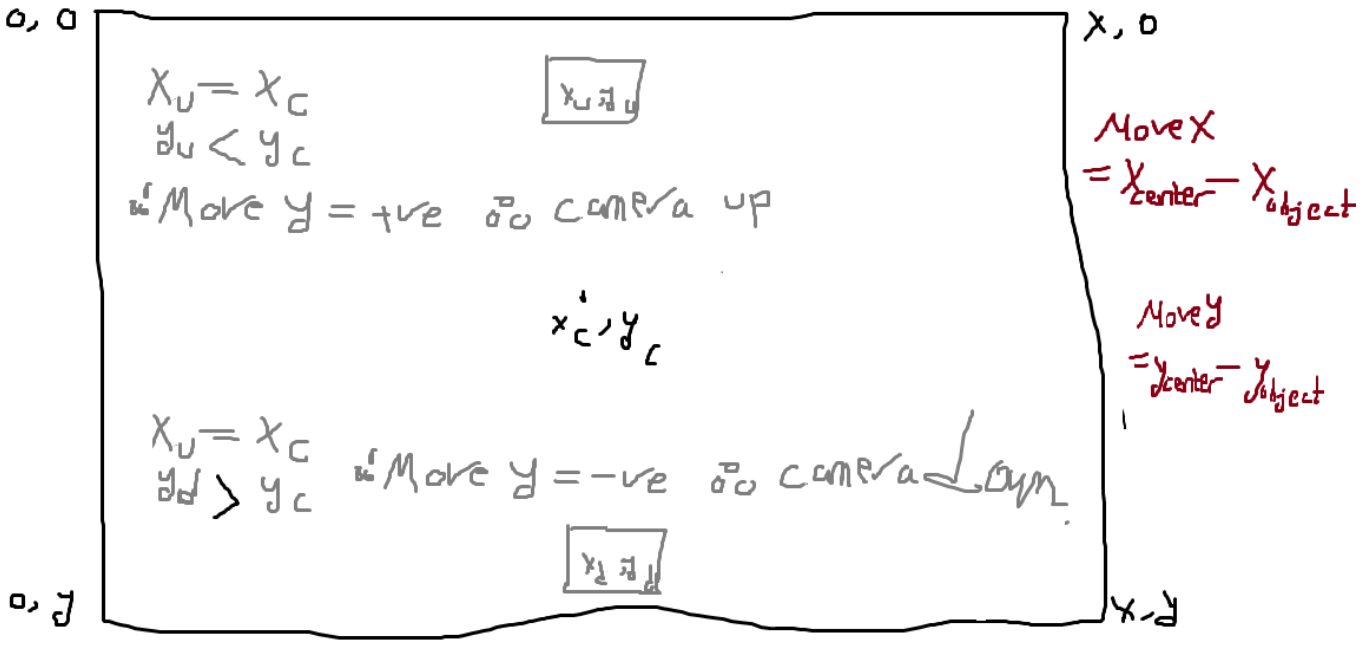
```
This function determines the movement instructions to guide the user in adjusting
the detected object towards the center of the frame
 Parameters: `frame`: Input frame containing the detected object
        `frameX`: X-coordinate of the frame center.
        `frameY`: Y-coordinate of the frame center.
        `boxX`: X-coordinate of the detected object center.
        `boxY`: Y-coordinate of the detected object center.
Alogorithm Explanation:

Given the center point of the frame and the current position of the object in the
frame we can get the movement instructions as in the following images
```

X-AXIS:




Y-AXIS




Combining both conditions we can get the exact instructions to get to the center

**5-Public Method: 'Start'**

To start the algorithm

## How To Use It:

- Certainly train the model on your object you want to apply the algorithm to
- Make an instance of the class Estimator with the trained model path and your camera channel
- Start the algorithm

## Ideas to be added:

- Model be more generalized to more objects but this will result in a new problem which is we don't always know the length of the objects we have
- So we add an object meter algorithm to detect the object's length
- Mapping: using tracking coordinates we generate from the movements algorithms we can build a simple map using UI tools